REMARKS

Claims 1-7, 9-25, 27-37, and 39-47 are now pending in the application. Claims 1, 3, 4, 5, 6, 9, 13, 14, 16, 17, 18, 22, 37, 39, 40, 41, and 42 are amended. Claims 8 and 38 are cancelled. Claims 45 – 47 are added. No new matter has been added. Reconsideration and reexamination are respectfully requested in view of the amendments and the following remarks.

Specification and Drawings

Figure 2 is amended to remove reference numbers 212 and 216, which are not described in the specification. The specification is amended to have references to step 316 of Figure 3 and step 414 of Figure 4 where those steps are described. The reference to "step 30, FIG. 3" on page 14, line 23, is corrected to refer to "step 304" and a typographical error in the same sentence is corrected. Applicant respectfully submits that the specification and drawings are now in order for allowance.

Claim Objections

The informalities noted by the examiner are corrected. In particular, claims 1 and 37 are amended to recite "an interesting operation" rather than "an operation of a predetermined type." Claims 8 and 38 are cancelled, thereby mooting the examiner's objection as it refers to these claims. The claims are now consistent in using "an interesting operation." The applicant therefore respectfully submits that claims 3 and 4 are no longer objectionable for the reasons noted by the examiner.

Claim Rejections 35 USC § 112

Claims 1, 8, 37, and 38 stand rejected under 35 U.S.C. 112, first paragraph. The examiner maintains that the specification does not enable these claims because the specification "does not enable any person skilled in the art ... to make or use the invention commensurate in scope with these claims," since the claim limitation "an operation of predetermined type" is not supported or found in applicants' specification.

Claims 8 and 38 are cancelled, thereby mooting the examiner's rejection of them.

Claims 1 and 37 are amended to recite "an interesting operation" in place of "an operation of predetermined type." The claim limitation "an interesting operation" is found and explained in the specification at page 7, lines 18-24, and page 8, lines 14-23. The claim limitation "an interesting operation" is now also explicitly defined in claims 1 and 37 to be "an operation by a user that changes the state of the document." Accordingly, applicant respectfully submits that claims 1 and 37 are enabled by the specification and are in condition for allowance.

Claim Rejections 35 USC § 103

The Examiner rejected claims 1-25 and 27-44 under 35 U.S.C. § 103(a) over Bristor (U.S. 6,018,342).

Claim 1 recites "a method implemented in a computer program application that performs operations on documents having states." The method includes "maintaining in a memory a state history of a document for storing document states" and, "whenever an interesting operation has occurred," "automatically capturing the state of the document as it exists after the operation and adding the captured state to the state history." As amended, an interesting operation is defined to be "an operation by a user that changes the state of the document."

Bristor does not disclose or suggest "maintaining in a memory a state history of a document for storing document states." Bristor also does not disclose or suggest, "whenever an interesting operation has occurred" "automatically capturing the state of the document as it exists after the operation and adding the captured state to the state history" where an interesting operation is "an operation by a user that changes the state of the document."

Bristor provides a method for automatically organizing user-generated signals – not document states. In Bristor's method, user-generated signals such as Unix shell commands (col. 3 lines 20-35) or Internet URLs (col. 4, lines 5-27) are converted into "user data" that convey information to a computer process (col. 11, line 67 to col. 12, line 8). Such user data may cause a processor to return, for example, a list of files in a directory or an image of a web page. These user-generated signals are accumulated in a history database. The user can, thereafter, take advantage of the history database to find and re-execute a particular user-generated signal.

Bristor's method allows the user to find a signal generated earlier in the user's history and re-execute it. Bristor does not, however, describe "automatically capturing *the state of the document* as it exists after the operation." Rather, Bristor is directed to capturing *user-generated signals*. Bristor also does not describe "maintaining in a memory *a state history of a document*." Rather, Bristor describes the creation of *a history of user commands*.

Bristor also does not suggest "automatically capturing the state of the document as it exists after the operation" or "maintaining in a memory a state history of a document." It would not have been obvious, based on Bristor, to "automatically captur[e] the state of the document as it exists after the operation" or "maintain[] in a memory a state history of a document" because Bristor describes a method for re-executing commands, and those commands are executed on the *current* state of documents – they do not recover the *previously existing* state of a document.

Because at least one element of the claimed invention is not taught or suggested by Bristor, no prima facie case of obviousness under 35 U.S.C. § 103 has been established. Accordingly, the applicant submits that claim 1 is allowable.

Claims 2 – 7 depend from claim 1 and are allowable for at least the same reasons as claim 1.

Claim 3 is amended to recite the method of claim 1, further comprising "maintaining in the state history the order in which the stored states were automatically added to the state history" and "displaying the state history to a user as a list of document states shown in their stored order."

As discussed in reference to claim 1, Bristor describes a method to "regenerate previously generated user data," as for Unix shell commands and the internet browsers, where "user data" are user-generated signals – not states of a document (col. 11, lines 49-60 and Fig. 5B, step 512). Thus, the history list described by Bristor is a history of user-generated signals, to be executed on the current state of all documents. Bristor does not, therefore, describe or suggest "displaying the state history to a user *as a list of document states* shown in their stored order." Accordingly, claim 3 is allowable.

Claim 4 is amended to recite the method of claim 3, wherein "the list of document states displayed to the user comprises a list of items, each item representing a state of the document that existed after an interesting operation and that can be recovered directly by selecting the item." Claim 4 is allowable for the reasons discussed for claim 3. Additionally, Bristor describes following a series of hyperlinks to retrieve various documents – not states of "*the* document" – and then stepping back through that series of links to revisit those documents (col. 4, lines 28-49). Thus, Bristor does not describe "a list of items, each item representing a state of the document that existed after an interesting operation and that can be recovered directly by selecting the item." Accordingly, claim 4 is allowable.

Claim 5 is amended to recite the method of claim 4, further comprising "providing a tool operable under user control to obtain source material from any state in the state history and apply it to a current state of the document, where the document is a raster image. Claim 6 is amended to recite the method of claim 4, further comprising "enabling a user to select any item in the displayed list of items and cause the application to create a new document having the document state corresponding to the selected item." Claims 5 and 6 are also allowable for the reasons discussed for claims 3 and 4.

Claim 7 recites the method of claim 4, wherein "each of the captured states in the state history maintains the state data in essentially its original form." Bristor describes a history list where repeated references to the same document are avoided by "removing from the history list links which are subsequently followed in reverse by use of the "Back" command." (col. 4, lines 56-59). Bristor does not describe or suggest "a list of document states" where "each item represent[s] a state of *the* document" and, because there is no description of "a *state* history of *a* document," there is no description and no suggestion to "maintain the state data in essentially its original form."

Claim 8 is cancelled, thereby mooting the examiner's rejection.

Claim 9 is amended to recite a method including "receiving from the user a sequence of commands to change the document," "changing the document state in response to each command," "adding the changed document state to a state history maintained in a computer-readable memory device each time the document state is changes" and, "for each document state added to the state history, adding a corresponding entry to a history list displayed to the user on a computer-controlled display device operated as part of a graphical user interface," and "in response to a user action selecting an item in the history list and establishing the document state corresponding to the selected item in the history list as the current state of the document." The applicant submits that claim 9 is allowable for the reasons given for claims 1 and 6.

Claims 10-15 depend from claim 9 and are allowable for at least the same reasons as claim 9.

Claim 13, as amended, recites the method of claim 9, further comprising "in response to a user command to change the document state corresponding to the selected item in the history list and established as the current state of the document, *deleting* the items after the selected item in the history list and the corresponding document states from the state history."

As noted with respect to claim 7, Bristor describes a method in the Netscape Web browser for stepping back through previously viewed documents and not listing redundant user data (col. 4, lines 5-19 and 56-65). But Bristor does not describe "a user command to change the document state" of any of those documents. Thus, there is no suggestion to "delet[e] the items after the selected item in the history list and the corresponding document states from the state history" "in response to a user command to change the document state." Accordingly, claim 13 is allowable over Bristor.

Claim 14, as amended, recites the method of claim 9, further comprising "in response to a user command to change the document state corresponding to the selected item in the history list and established as the current state of the document, *maintaining* the items after the selected item in the history list and *adding* a new item to the end of the history list and a new document state to the corresponding document states from the state history." Claim 14 is allowable for the same reasons that claim 13 is allowable. That is, because Bristor does not describe "a user command to change the document state" of any of those documents, there is no suggestion to, "in response to a user command to change the document state ... maintain[] the items after the selected item in the history list and add[] a new item to the end of the history list and a new document state to the corresponding document states from the state history."

Claim 16 is amended to recite a method that includes "maintaining in a memory a state history of a document," "in response to a user action, selecting a first state from the state history and establishing the first selected state of the document as the current state of the document," "in response to a user action, selecting a second state from the state history, the second state being a state created after the first state, as a source of data for an operation," and "performing the operation with the data from the second state on the first state." Applicant respectfully submits that claim 16 is allowable for the reasons discussed for claim 1.

Additionally, Bristor does not describe or suggest "selecting a second state from the state history, the second state being a state created after the first state," or "performing the operation with the data from the second state on the first state." Bristor describes a Back command which, "when issued by the user, retrieves and displays the Web document which immediately follows the currently displayed Web document in the history list, which is organized in reverse chronological order" (col. 4, lines 15-23). However, the Back command merely re-retrieves and displays a Web document in its current state; it does not "establish the first selected state of the document as the current state of the document." Even if it did, Bristor does not describe "selecting a second state" where the second state was "created after the first state." Indeed, there could be no state between the current state and the previous state. Bristor therefore does not suggest the method described in claim 16. Finally, the forward command described by Bristor does not "perform[] the operation with the date from the second state on the first state." (col. 4, lines 23-27) Rather, the forward command selects the next user-generated signal in the history list and executes it. Accordingly, claim 16 is allowable over Bristor.

Claim 17, as amended, recites a method that includes "keeping a history of document states of a document, the document states being created automatically whenever a user command to the application changes the state of the document and being complete in themselves," "enabling the user to discard any of the states in the history to create a revised history," and "enabling the user to step backward and forward through the revised history and thereby alter the state of the document to be any of the document states in the revised history.

Applicant respectfully submits that claim 17 is allowable for the reasons discussed for claim 1. Bristor does not describe "a history of document states of a document...created whenever a user command to an application changes the state of *the document*." Rather, Bristor describes a history mechanism for a World Wide Web browser, including "a list of some of the most recently retrieved Web documents" (col. 3, lines 32-52, 56-65).

Claim 17 is also allowable for the following reason. Assuming that the user-generated signals described by Bristor were directed to editing a single document and making successive changes in the document state, it might be possible to recreate the earlier state of the document by, for example, undoing the effect of each user-generated signal. However, if this were the case, it would not be possible to "discard any of the [signals] in the history" because so doing would make it impossible to "alter the state of the document to be any of the document states in the revised history." Indeed, the method of claim 17 is possible because the document states are "complete in themselves," such that some may be deleted with no adverse effect (specification, page 12, lines 1-3). Thus, Bristor does not describe or suggest "enabling the user to discard any of the states in the history to create a revised history" or "enabling the user to step backward and forward through the revised history and thereby alter the state of the document to be any of the document states in the revised history," and claim 17 is allowable.

Claim 18, as amended, recites a method that includes "keeping a history of document states of a document, the document states being created automatically whenever a user command to the application changes the state of a document and begin complete in themselves," "enabling the user to discard any of the states in the history to create a revised history," and "enabling the user to designate any one of the document states in the revised history and thereby establish the designated state as the current state of the document." Applicant respectfully submits that claim 18 is allowable for the reasons discussed for claim 1. Claim 18 is also allowable for the reasons discussed for claim 17.

Claims 19-21 depend from claim 18 and are allowable for at least the reasons given for claim 18.

Claim 22 recites a method including "identifying for the user on a display device a set of states that the document has been in by operation of the application" and "enabling the user to designate any one of the identified states as a document state operand." Claim 22 is allowable for the reasons discussed above for claims 1.

Claims 23-25, 27-36, and 44 incorporate the features of claim 22 and are allowable for at least the same reasons given for claim 22.

Claim 37 is an apparatus claim corresponding to claim 1. Claim 37 is allowable for at least the reasons set forth above in connection with claim 1.

Claim 38 is cancelled, thereby mooting the examiner's rejection.

Claim 39 an apparatus claim corresponding to claim 9. Claim 39 is allowable for the reason given for claim 9.

Claim 40 is a computer program claim corresponding to claim 16. Claim 40 is allowable for the reason given for claim 16.

Claims 41and 42 are computer program claims corresponding to claims 17 and 18, respectively. Claims 41 and 42 are allowable for the reasons given for claim 17 and 18.

Claim 43 is allowable for at least the reasons given for claims 1.

Conclusions

Attached is a marked-up version of the changes being made by the current amendment.

Applicant submits that all of the claims are now in condition for allowance, which action is requested. No fee is calculated to be due with this response. Please apply any charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: 7 | 24 | 2002

[illegible]
Tamara Fraizer
Reg. No. P-51,699

Fish & Richardson P.C.
500 Arguello Street, Suite 500
Redwood City, California 94063
Telephone: (650) 839-5070
Facsimile: (650) 839-5071

50098487.doc

Version with markings to show changes made

In the specification:

Please replace the paragraph beginning at page 13, line 8 with the following rewritten paragraph:

As shown in FIG. 3, the operation of an implementation of the state history feature may be seen to begin when a user enters a command (step 302). A command is a request made by the user to the application, through a graphical user interface, for example, to change the state of the application or the state of the active document. The application performs the corresponding action (step 304, which will be elaborated in reference to FIG. 4). If the action is one categorized as not affecting the state of a document (the no branch of decision step 306), the state history and history list are unaffected. Otherwise, if the action of the command was performed with the document in the state of the temporally most recent entry in the history list, which corresponds to the temporally most recent state in the state history, the state of the document after the action is stored in the state history and the history list on the history palette is updated (the yes branch of step 308, and steps 312, 314, and 320). On the no branch of step 308, if non-linear history mode is in effect, nothing is discarded from the history list or the state history merely because an interesting command has been entered (the yes branch of step 310). In linear history mode (the no branch of step 310), the history list items below the current state item and the corresponding states in the state history are discarded (step 316), other than any item selected as a source for history-based tools, such as the history paintbrush (see description of buttons 210, above).

Please replace the paragraph beginning at page 14, line 12 with the following rewritten paragraph:

To produce a practical implementation, it is advantageous to store document states in a form that allows a great deal of sharing between saved states, to keep to a minimum the amount of memory consumed in storing redundant information, and the processor resources consumed in writing and reading it. A data representation suitable for this purpose is described in commonly-owned U.S. patent [application serial] no. [08/702,941] 5,905,506 to Hamburg for "Shared Tile

Image Representations" [filed August 26, 1996], the disclosure of which is incorporated here by this reference.

Please replace the paragraph beginning at page 14, line 22 with the following rewritten paragraph:

Referring to FIG. 4, aspects of the perform action step (step 304, FIG. 3) pertinent to an implementation of a state history feature will no[t]w be described. If the user's command is not an undo or a redo, that is, not the Ctrl-Z toggle, the command is stored in an undo buffer associated with the active document (step 414), such as buffers 134 and 144 of FIG. 1 (the yes branch of decision step 410 and step 412). If the command is an undo or a redo, the undo or redo operation of the command in the undo buffer is executed (step 412). It should be noted that the history-related commands, and in particular changes to the history list and the state history, can be subject to the conventional undo and redo (step 412). The provides an advantageous and elegant user interface that allows, for example, a user to toggle between two document states previously selected from anywhere in the state history with a single repeated keystroke.

In the claims:

Claims 1, 3, 4, 5, 6, 8, 9, 13, 14, 16, 17, 18, 22, 37, 38, 39, 40, 41, and 42 has been amended as follows:

1. (Twice amended) A method implemented in a computer program application performing operations on documents having states, the method comprising:

maintaining in a memory a state history of a document [for storing document states]; and

whenever an [operation of a predetermined type] interesting operation has occurred, an interesting operation being an operation by a user that changes the state of the document, automatically capturing the state of the document as it exists after the operation and adding the captured state to the state history of the document.

2. The method of claim 1, wherein the memory comprises a disk file.

3. (Amended) The method of claim 1, [wherein] further comprising:

[the state history includes states of the document and] maintaining in the state history the order in which the stored states were automatically added to the state history; and

displaying the state history [is displayed] to a user as a list of document states shown in their stored order[;].

[an operation is classified as an interesting operation if it changes the state of the document;

a state is added to the state history only if the operation creating the state is classified as an interesting operation and not otherwise;

performing a step backward operation by installing as the current state of the document a state stored in the state history, whereby all step backward operations place the document in a state that occurred immediately after an interesting operation;

performing a step forward operation by installing as the current state of the document a state stored in the state history, whereby all step forward operations place the document in a state that occurred immediately after an interesting operation;]

4. (Amended) The method of claim 3, wherein:

[the list of document states displayed to the user comprises a list of items, each item representing a state of the document that existed after an interesting operation and that can be recovered with a step backward operation in the application; and]

the list of document states displayed to the user comprises a list of items, each item representing a state of the document that existed after an interesting operation and that can be recovered directly by selecting the item.

5. (Amended) The method of claim 4, [wherein] further comprising:

[the application is digital graphics program operable to create and revise images in digital form;]

[the application provides] providing a tool operable under user control to obtain source material from any state in the state history and apply it to a current state of the document[; and the images are], where the document is a raster image[s].

6. (Amended) The method of claim 4, [wherein] further comprising:

[the application enables] enabling a user to select any item in the displayed list of items and cause the application to create a new document having the document state corresponding to the selected item.

7. The method of claim 4, wherein:

each of the captured states in the state history maintains the state data in essentially its original form, whereby the captured state data is suitable for immediate use in other operations.

8. (Cancelled)

9. (Amended) A computer-implemented method of interacting with a user editing a document in a computer program application, the document having a document state, the method comprising:

receiving from the user a sequence of commands to change the document;

changing the document state in response to each command;

adding the changed document state to a state history maintained in a computer-readable memory device each time the document state is changed;

for each document state added to the state history, adding a corresponding entry to a history list displayed to the user on a computer-controlled display device operated as part of a graphical user interface; and

in response to a user action [stepping backward to], selecting an item in the history list [, establishing the document state corresponding to the selected item in the history list as the current state of the document.

10. The method of claim 9, wherein:

the state history and the history list are limited to storing a preset number of items and excess items are scrolled off the top of the list as new items are added.

11. The method of claim 9, wherein:

the state history is stored in a region of memory and the oldest document states in the state history are discarded when free space in the region runs low.

12. The method of claim 11, wherein:

the oldest document states are found and discarded by a memory management process.

13. (Amended) The method of claim 9, further comprising [wherein]:

in response to a user [a] command to change the document state corresponding to the selected item in the history list and established as the current state of the document, [that comes after a step backward command to a selected item in the history list causes] deleting the items after the selected item [to be deleted from] in the history list and the corresponding document states [to be deleted] from the state history.

14. (Amended) The method of claim 9, further comprising [wherein]:

in response to a user [a] command to change the document state corresponding to the selected item in the history list and established as the current state of the document, [that comes after a step backward command to a selected item in the history list does not cause] maintaining the items after the selected item [to be deleted from] in the history list and [adds] adding a new item to the end of the history list and a new document state to the state history.

15. The method of claim 9, further comprising:

enabling a user interface gesture on the history list to create a new document from a document state from the state history.

16. (Amended) A method implemented in a computer program application operable to create and edit a document, comprising:

[keeping a history list] maintaining in a memory a state history of a document;

in response to a user action, [going back to a previous] selecting a first state [in] from the state history [list] and establishing the first selected state of the document as the current state of the document;

in response to a user action, selecting a [future] second state from the state history [list], the second state being a state created after the [previous] first state, as a source of data for an operation; and

performing the operation with the [future] data from the second state on the [previous] first state.

17. (Amended) A method implemented in a computer program application operable to create and edit a document, comprising:

keeping a history of document states [created by a user]of a document, the document states being created automatically whenever a user command to the application changes the state of the document and being complete in themselves;

enabling the user to discard any of the states in the history to create a revised history; and enabling the user to step backward and forward through the revised history and thereby [to] alter the state of the document to be any of the document states in the revised history.

18. (Amended) A method implemented in a computer program application operable to create and edit a document [document-processing computer program application, the method] comprising:

keeping a history of document states of a document, the document states being created automatically whenever a user command to the application changes the state of [a]the document and being complete in themselves;

enabling the user to discard any [user-selected set] of the [document] states in the history to create a revised history; and

enabling the user to designate any one of the document states in the revised history and thereby [install] establish the designated state as the current state of the document.

19. The method of claim 18, further comprising:

saving the history when the document is closed on a long-term storage medium, whereby the history may be restored when the document is later opened and across invocations of the application.

20. The method of claim 19, wherein:

the saved history resides in the document with final document data.

21. The method of claim 19, wherein:

the saved history resides in a long-term data repository independent of the original document.

22. (Thrice amended) A method enabling a user to control operation of a computer program application for creating and modifying a document, the method comprising:

identifying for the user on a display device a set of states that the document has been in by operation of the application; and

enabling the user to designate any one of the identified states as a document state operand[; and

providing the user an editing tool having a document state operand derived from the designated state].

23. The method of claim 22, further comprising:

displaying the document in a user interface window, the document being a digital image.

24. The method of claim 23, wherein the digital image has a plurality of layers, each of the plurality of layers having a plurality of channels, the method further comprising:

displaying user-interface elements for applying filters to the digital image.

25. The method of claim 22, further comprising:

[installing] establishing the designated state as the current state of the document in response to a user command.

26. (Cancelled)

27. (Previously amended) The method of claim 22, further comprising:

providing the user a delete tool for deleting the designated state from the set of states.

28. The method of claim 22, wherein:

the set of states is identified by displaying a scrollable list of elements each identifying one of the states in the set.

29. The method of claim 28 wherein the list elements are ordered by the time the corresponding states were created.

30. The method of claim 25 wherein the designation and [installation] <u>establishment</u> are performed in response to a single command.

31. The method of claim 25 wherein the set of states is displayed in an order, the method further comprising:

enabling the user to make a gesture on a user interface indicating a sequence of displayed state identifiers and responding to the gesture by displaying the document in the states indicated as the gesture is made.

32. The method of claim 25 further comprising:

enabling the user to modify the document state after the [installing] <u>establishing</u> step; and

adding the document state resulting from the modification to the set of states identified on the display device.

33. The method of claim 31 wherein the set of states is displayed in order of creation of the states in the set.

34. The method of claim 31 wherein the document is a digital image.

35. The method of claim 25 further comprising:

providing a step backward and a step forward command for the user to execute to navigate the set of states; and

providing a separate undo and redo command for the user to undo and redo commands entered by the user.

36. (Previously amended) The method of claim 22, further comprising:

providing a step backward and a step forward command for the user to execute to navigate the set of states; and

providing a separate undo and redo command for the user to undo and redo commands entered by the user.

37. (Twice amended) Apparatus comprising a computer-readable storage medium tangibly embodying program instructions defining a computer program application for performing operations on documents having states, the program comprising instructions operable for causing a programmable processor to:

maintain in a memory a state history of a document [for storing document states]; and

whenever an interesting operation [of a predetermined type] has occurred, an interesting operation being an operation by a user that changes the state of the document, the state being complete in itself and independent of other states; automatically capture the state of the document as it exists after the operation and add[ing] the captured state to the state history of the document.

38. (Cancelled)

39. (Amended) Apparatus comprising a computer-readable storage medium tangibly embodying program instructions for interacting with a user editing a document in a computer program application, the document having a document state, the apparatus comprising instructions operable for causing a programmable processor to:

receive from the user a sequence of commands to change the document;

change the document state in response to each command;

add the changed document state to a state history maintained in a computer-readable memory device each time the document state is changed;

for each document state added to the state history, add a corresponding entry to a history list displayed to the user on a computer-controlled display device operated as part of a graphical user interface; and

in response to a user action [stepping backward to], select an item in the history list [, update the document to have the corresponding document state saved in the state history] and establish the document state corresponding to the selected item in the history list as the current state of the document.

40. (Amended) A computer program, residing on a computer-readable medium, comprising instructions for causing a computer to:

keep a history list;

in response to a user action, [go back to a previous] select a first state [in] from the history list and establish the first selected state of the document as the current state of the document;

in response to a user action, select a [future] second state from the history list, the second state being a state created after the [previous] first state, as a source of data for an operation; and

perform the operation with the [future] data from the second state on the [previous] first state.

41. (Amended) A computer program, residing on a computer-readable medium, comprising instructions for causing a computer to:

keep a history of document states created by a user; the document states being created automatically whenever a user command to the application changes the state of a document and being complete in themselves;

enable the user to discard any of the states in the history to create a revised history; and enable the user to step backward and forward through the revised history and thereby [to] alter the state of the document to be any of the document states in the revised history.

42. (Amended) A computer program, residing on a computer-readable medium, comprising instructions for causing a computer to:

keep a history of document states created by a user, the document states being created automatically whenever a user command to the application changes the state of a document and being complete in themselves;

enable the user to discard any [user-selected set] of the [document] states in the history to create a revised history; and

enable the user to designate any one of the document states in the revised history and thereby [install] establish the designated state as the current state of the document.

43. (Previously amended) A computer program, residing on a computer-readable medium, comprising instructions for causing a computer to:

create and modify a document;

identify for a user on a display device a set of states that the document has been in by operation of the application; and

enable the user to designate any one of the identified states.

44. The method of claim 36, further comprising:

providing to the user a first undo command function that operates with reference to the first history and a second undo command function that operates with reference to the second history.

45. (New) The method of claim 3, further comprising:

[installing] <u>establishing</u> as the current state of the document a state stored in the state history.

46. (New) The method of claim 1, further comprising:

maintaining in memory a history of all operations requested by a user, including operations global to the state of the application.

47. (New) The apparatus of claim 37, further comprising instructions operable for causing a programmable processor to:

maintain in memory a history of all operations requested by a user, including operations global to the state of the application.